PR #39524 完整报告

vllm-project/vllm

[Refactor] Remove `resampy` dependency

合并时间：2026-04-16 23:48

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/39524

PR 分析报告：移除 resampy 依赖

执行摘要

本 PR 移除 resampy 音频重采样依赖，默认改用 pyav 方法，基于性能基准测试显示 pyav 在速度和质量上显著优于 resampy。变更涉及多个音频模块文件和依赖配置，简化安装并优化处理流程，但引入潜在运行时错误风险，需关注音频功能稳定性。

功能与动机

PR 的动机源于性能优化：作者在 PR body 中提供数据，表明 pyav 在音频重采样上平均耗时 10.847 ms，而 resampy 为 443.071 ms，且质量指标（如 RMSE）更优。因此，移除不必要的 resampy 依赖，以提升系统效率和简化依赖管理。

实现拆解

1. 入口变更：从 vllm/multimodal/audio.py 开始，删除 resampy 的导入和 resample_audio_resampy 函数。
2. 核心逻辑改造：更新 AudioResampler 类，将默认方法从 "resampy" 改为 "pyav"，并移除 "resampy" 选项，确保重采样仅支持 pyav 和 scipy。python class AudioResampler: def __init__(self, target_sr: float | None = None, method: Literal["pyav", "scipy"] = "pyav"): self.target_sr = target_sr self.method = method # 默认切换到 pyav def resample(self, audio, *, orig_sr): if self.method == "pyav": return resample_audio_pyav(audio, orig_sr=orig_sr, target_sr=self.target_sr) elif self.method == "scipy": return resample_audio_scipy(audio, orig_sr=orig_sr, target_sr=self.target_sr) else: raise ValueError("Invalid resampling method")
3. 媒体音频调整：在 vllm/multimodal/media/audio.py 中，移除 resampy 导入，并将 load_audio_soundfile 中的重采样调用改为 resample_audio_pyav。python def load_audio_soundfile(path, *, sr=None, mono=True): if sr is not None and sr != native_sr: y = resample_audio_pyav(y, orig_sr=native_sr, target_sr=sr) # 直接使用 pyav return y, int(sr)
4. 依赖配置更新：修改 setup.py，从 audio extras 移除 resampy；同步清理测试要求文件（如 requirements/test/cuda.in）。
5. 测试配套：运行 pytest -s -v tests/entrypoints/openai/correctness/test_transcription_api_correctness.py 验证功能正确性，测试通过确保无回归。

vllm/multimodal/audio.py

音频重采样核心逻辑文件，移除 resampy 导入和函数，更新 AudioResampler 默认方法，直接影响音频处理功能。

```
class AudioResampler:
    """Resample audio data to a target sample rate."""

    def __init__(
        self,
        target_sr: float | None = None,
        method: Literal["pyav", "scipy"] = "pyav", # 变更：默认方法从 "resampy" 改为 "pyav"，并移除 "resampy" 选项
    ):
        self.target_sr = target_sr
        self.method = method

    def resample(
        self,
        audio: npt.NDArray[np.floating],
        *,
        orig_sr: float,
    ) -> npt.NDArray[np.floating]:
        if self.target_sr is None:
            raise RuntimeError(
                "Audio resampling is not supported when `target_sr` is not provided"
            )
        if math.isclose(float(orig_sr), float(self.target_sr), rel_tol=0.0, abs_tol=1e-6):
            return audio
        if self.method == "pyav":
            return resample_audio_pyav(audio, orig_sr=orig_sr, target_sr=self.target_sr)
        elif self.method == "scipy": # 变更：移除 "resampy" 分支，只保留 "pyav" 和 "scipy"
            return resample_audio_scipy(audio, orig_sr=orig_sr, target_sr=self.target_sr)
        else:
            raise ValueError(
                f"Invalid resampling method: {self.method}. "
                "Supported methods are 'pyav' and 'scipy'." # 错误消息同步更新
            )
```

vllm/multimodal/media/audio.py

音频加载媒体文件的关键模块，调整重采样调用以使用 pyav，影响音频输入处理路径。

```
def load_audio_soundfile(
    path: BytesIO | Path | str,
    *,
    sr: float | None = 22050,
    mono: bool = True,
) -> tuple[np.ndarray, int]:
    """Load audio via soundfile"""
    with soundfile.SoundFile(path) as f:
```

```
            native_sr = f.samplerate
            y = f.read(dtype="float32", always_2d=False).T

        if mono and y.ndim > 1:
            y = np.mean(y, axis=tuple(range(y.ndim - 1)))

        if sr is not None and sr != native_sr:
            y = resample_audio_pyav(y, orig_sr=native_sr, target_sr=sr) # 变更：从 resampy.resample
            切换到 pyav 实现
            return y, int(sr)
        return y, native_sr
```

评论区精华

review 讨论聚焦于正确性风险：

> gemini-code-assist[bot] 评论："In load_audio_soundfile, the resampling logic now hardcodes resample_audio_pyav. If the av package is not installed, this will raise an error... It might be safer to use the AudioResampler class or provide a fallback to scipy."

此评论指出硬编码 pyav 可能导致运行时错误，但未在 PR 中进一步解决，反映设计权衡中性能优化与鲁棒性的冲突。

风险与影响

- 技术风险：若用户未安装 av（如省略 audio extras），音频重采样将失败；移除 resampy 可能影响依赖旧方法的代码；默认方法变更可能引入质量不一致性。
- 影响范围：用户需确保 av 可用以享受性能提升；系统减少依赖复杂度；团队维护简化，但需监控音频功能稳定性，尤其是边缘案例和多平台支持。

关联脉络

从历史 PR 看，PR 39997 将 pyav 和 soundfile 移到基础依赖，与本 PR 共同优化音频处理依赖链，显示团队在简化安装和提升性能上的持续努力。结合近期 PR 如音频模型支持（PR 39575），可看出多模态功能演进中依赖管理的系统化调整。